Europäisches Patentamt
European Patent Office
Office européen des brevets

Abstract of DE10008008	Print	Copy	Contact Us	Close

Result Page

Explained one becomes among other things a processor for the treatment of interrupt requests and events, which contains several functional units, which work parallel on various instructions or on various parts of an instruction. In addition the processor contains an interruption processing unit, those the processing of interrupt requests and/or. Events permits or closes. Become over a test unit selected permit the mode "interrupt request".

▲ top

Result Page

The invention relates to a processor for the processing of interrupt requests. The processor contains functional units, the temporal parallel and/or. temporal overlapping at various instructions or at various parts of an instruction works. These functional units become also pipelines called. In addition the processor contains an interruption processing unit, which permits the processing of interrupt requests in a first mode and which in a second mode the processing closes from interrupt requests. Interrupt requests are requests, which require an interruption of the work of the functional units. The interrupt requests become also designated as interrupts.

There are interrupt requests, which become generated from the outside, so called asynchronous interruptions. In addition there are interrupt requests, which become generated with the execution of an instruction within the processor, so called synchronous interruptions.

A processor contains usually on a chip an arithmetic unit and a control mechanism. Known processors, z. B. the processor Pentium III, contain a status bit of the company Intel to the switching of the mode. The status bit is part of a status word, which becomes also designated as processor status word or machine status word. Changes in the status word usually lead to a termination of the parallel and/or. temporal overlapping processing of the instructions which can be implemented. Only after the processor has the status changes processed, becomes again with the parallel and/or. overlapping instruction execution started.

From the EP 0586 847 A1 an arrangement is described to the processing of interrupt requests. The dynamics of such arrangements depend to a large extent on the ability to be able to work on tasks of system with differing urgency. The processing of the tasks of system becomes general performed by interruption-controlled processes. In order to avoid a dynamic reduction with the frequent use of an interrupt control for the processing of a chain from administrative routines to, changed becomes with the described arrangement with tasks of system working on processes, from the interruption-controlled mode into a cyclic query-controlled mode.

It is object of the invention to indicate an improved processor for the treatment of interrupt requests.

This object becomes 1 dissolved by a processor with the features of the claim. Developments are in the Unteransprüchen indicated.

The invention is the basis the consideration that usually the processing of interruptions becomes allowed during the normal execution of programmes. However the processing other interruptions becomes disabled during the execution from programmes to the treatment of an interrupt request. The interruptions become thereby dependent disabled of a status bit allowed or. The status bit becomes very rare switched in the comparison the number of the instructions which can be implemented, for example only all ten thousand instructions. However there is applications, with those the switching many more frequent, z. B. after the execution of only five to ten instructions required would be. Such an application is for example the emulation of an origin processor with the execution of a programme written for the origin processor by a target processor, whose structure differs from the structure of the origin processor. While the emulation of a single instruction of the origin processor no interruption may take place, in order to avoid inconsistency in the states of the origin processor copied during the emulation. On the other page also asynchronous interruptions may be not to prolonged disabled, if possible by the emulation a similar processing of interruptions is to become as on the origin processor. Therefore an interruption in too large distances, z does not have. B. after the emulation of in each case five instructions of the origin processor allowed become. With other words the mode would have to become frequent switched, in order to make processing possible from interruptions to. The interruptions arise however, as before also, to only relative rare.

The invention proceeds further from the consideration that the advantages of a temporal parallel and/or. any longer do not arise to temporal overlapping processing of instructions or of various parts of an instruction if the processing must become continuous interrupted. In order to use these advantages nevertheless, tested becomes in the processor according to invention in the second mode whether an interrupt request is to be worked on. Dependent one of the test result becomes only connected when being present interrupt requests into the first mode.

The test unit possible it, without switching a previous state change of the processor status if necessary into the first mode. A state change of the processor status would require an instruction, would have to become emptied before whose execution the pipelines, since only with the instruction execution in the processor determined becomes, which status change arises and is like serious these.

Only if during examining found becomes that interruptions arose, the parallel becomes and/or. overlapping execution of instructions with the switching into the first mode interrupted.

The invention can be begun therefore in particular if a change between inhibition and permission of the processing of interruptions were frequent required. Thus there are also different applications, than the applications specified in this application. Straight one with applications, in which frequent between the first mode and the second mode would have to become changed, becomes with the processor according to invention a significant higher processing speed too these applications belonging programmes achieved than with known processors. The parallel execution of instructions becomes interrupted only if actual interrupt requests are to be worked on. Accomplishing the test unit does not lead against it yet to an interruption of the parallel instruction execution.

The measures according to invention can be begun at location of the known proceedings to the selection of the mode or in combination with these proceedings.

With a development of the processor according to invention the interrupt requests processed by the interruption processing unit are caused by processor -external procedures. With other words are this asynchronous interrupt requests, which become for example by another processor same type in a multiprocessor system, caused by in/expenditure components or by signals from the outside. This embodiment is particularly favourable, if programmes executed to become to be supposed, would be very frequent required with which without the invention a change-over of the mode. Such an application is the emulation, like already above detailed explained.

With a next development the processor contains a memory unit to memories of the interrupt requests in the second mode. Such a memory unit could become for example according to the queue principle constructed, with which the interrupt requests arising in the second mode are incorporated into a queue. Only with the later switching into the first mode, the interrupt requests become in sequence processed.

With another embodiment the processor is so constructed that the test unit becomes triggered by a particular instruction of the command sentence. If interrupt requests are in the queue with the execution of the particular instruction, then these processed become.

Another possibility for releasing the test unit is using an indicator in at least an instruction word. The indicator is for example a certain bit position. If the bit in this place has the value "1", then the test unit becomes executed. With the value "0" no test unit executed becomes.

With an embodiment the test unit becomes triggered with set indicator before and after the execution of the instruction, which contains the indicator.

▲ top

Particularly with processors with VLIW architecture (Very Long Instruction Word), with which the instruction word several instructions contains, can an indicator be favourably used, since an additional bit is only required for a group of instructions of a VLIW.

The invention relates to for the treatment of events in addition a processor, which contain at least one control unit to the execution of instructions of a command sentence. A control unit becomes required, if instructions only successively or temporal partial overlapping executed to become to be supposed. Several control units are present with the parallel execution of instructions or parts of instruction.

By an event the switching state of the processor becomes changed. However the work of the control unit does not have to become compellingly interrupted in contrast to an interrupt request. Asynchronous and synchronous events can be differentiated also with events. Asynchronous events are events, which are to due to procedures outside of the processor, for example on procedures in other processors of the same multiprocessor system, z. B. processor-spreading instructions. Synchronous events are events, which develop with the execution of an instruction in the processor.

The events become treated with known processors, as soon as they arise. This leads to the fact that in a multiprocessor system processor-spreading instructions only used to become limited to be able. In addition expensive programmes required are, in order to make possible cooperations of the processors.

The difficulties become still larger, if programmes of an origin multiprocessor system by target processors of a goal multiprocessor system executed to become to be supposed. For such an emulation the known processors are only conditional appropriate.

It is object of a second aspect of the invention to indicate an improved processor for the treatment of events.

This object becomes 4 dissolved by a processor with the features of the claim. Developments are in the Unteransprüchen indicated.

The invention in accordance with second aspect proceeds from the consideration that the main cause for those is to be above seen mentioned problems in a missing event processing unit, which creates a possibility, to retard the processing from events to. Therefore the processor according to invention contains an event processing unit, which permits the processing of events in a first mode and which in a second mode the processing of the events closes. In the second mode of the event processing unit is ensured that the processor implements instructions unimpaired by the event. The change of the modes can take place from the outside to predetermined times. Thus the operation of processors in a multiprocessor system can be co-ordinated.

With an embodiment the control unit of the processor contains several functional units, the temporal parallel and/or. temporal overlapping at various instructions of the command sentence or at various parts of an instruction works. Straight one with processors, which have so complex control units, leads an event processing unit with the use in a multiprocessor system to a significant simplification with the accordance of the operation of the processors one on the other.

The mode can be selected for example by changing a status bit in a status word to the control of the operation of the processor. However it has this with processors, with several control units to the parallel execution of various instructions or parts of instruction the disadvantage that must become briefly interrupted with a change of the status bit the parallel execution of instructions. Changing a status bit is however a simple technical measure, which does not require additional instruction in the command sentence. For many applications changing of a status bit and the interruption connected thereby are also acceptable.

With an alternative and/or. kummulativen embodiment become in the second mode tested whether an event is to be worked on. Dependent one of the test result becomes connected when being present an event into the first mode. In particular for example no status bit in a status word becomes changed, in which also states are registered, which require an interruption of the work of the functional units. The processor would have to then interrupt before the processing the status word of changing instruction the parallel processing, since it can determine only with the treatment of this instruction, which status bit changed is. It turns out that no interruption of the work of the functional units was required, then the interruption cannot be cancelled any longer.

With a development the events are caused by processor-external procedures. With other words it concerns asynchronous events. Such events are z. B.:

- the processor-spreading Invalidierung of a so called TI-Buffers (translation Lookaside Buffer), or
- Procedures in connection with the operation of a fast buffer (cache).

With a next embodiment the processor contains a memory unit to memories of event messages with the occurrence of events in the second mode. The processing of the events can become by memories in the memory unit up to the switching into the first mode the delayed. The delay time can be given by targeted releasing of the test unit, so that in particular processors in a multiprocessor system synchronized to become to be able.

The test unit becomes triggered with an embodiment by a particular instruction of the command sentence. This leads in particular with programmes, with which the mode would have to become otherwise frequent switched, to significant shorter programmes. An example for such programmes are Emulationsprogramme.

Alternative one becomes the test unit by an indicator in at least an instruction word triggered. With an embodiment all instructions contain the command sentence this indicator. The indicator is for example a certain bit position. With VLIW processors (Very Long Instruction Word) the instruction word contains of several instructions. The indicator concerns then several instructions simultaneous.

If the processor contains both an interruption processing unit and an event processing unit, then the test unit becomes both in the interruption processing unit and in the event processing unit triggered with an embodiment with the help of an instruction word. Thus for example the processing of asynchronous interrupt requests and asynchronous events can be released with the help of an instruction.

With a processor with interruption processing unit and event processing unit in each case two various indicators or a common indicator can be used for releasing the test units in the interruption processing unit and the event processing unit in one, several or all instruction words.

Becomes the processor according to invention for the treatment of events and/or. , then processor-spreading instructions can one of its embodiments in a multiprocessor system used be used, even if these must take regard on synchronization points in other processors of the multiprocessor system. By an appropriate choice of releasing the test units in the event processing unit it can be ensured that the operation of a processor does not become disturbed by the other processors.

In the following the two aspects of the invention become explained on the basis the accompanying designs. In it show:

Fig. 1 procedures when implementing a Emulationsprogramms,

Fig. 2 a processor for the treatment of interrupt requests and events,

Fig. 3 method steps if Unterbre do not chungsanforderungen processed to become to be able,

Fig. 4 method steps with the execution of an instruction by the processor,

Fig. 5 an instruction word for a VLIW processor,

Fig. 6 method steps if events processed do not become,

Fig. 7 the treatment of events of respective method of steps with the execution of an instruction by the processor,

Fig. 8 an instruction word for a VLIW processor, and

Fig. 9 the use two processors for the treatment of events with the execution memories of a management program.

Fig. 1 shows procedures when implementing an emulator program 10, whose instructions are processed by a target processor 50, its structure down on the basis the Fig. 2 more near explained becomes. The emulator program 10 implements instructions source program 12. The instructions source program are by the command sentence of an origin processor predetermined, its structure itself from the command sentence on the basis the Fig. 2 of explained target processor 50 differentiates between. The instructions source program 12 become 10 successively processed of the emulator program,

▲ top

see arrow 14. First the bit pattern of each instruction word becomes into its functional components decomposed, D. h. for example into the operation code, in register operands and direct operands. Subsequent one loads the emulator program 10 the values of the operands and implements by the operation code the certain operation. The result becomes for the other processing z. B. in working registers of the target processor stored. Afterwards the programme counters of the origin processor copied in the emulator program 10 incremented becomes.

During the emulation source program 12 the registered emulator program 10 the frequency of the execution of certain portions source program 12. If a program part becomes very often executed, then the emulator program 10 starts a translation program 16. When implementing the translation program 16 becomes the respective portion source program 12 18 translated into a goal instruction sequence, which contains instructions of the command sentence for the target processor, sees arrows 20 to 24. There is not for each instruction source program 12 a corresponding instruction in the command sentence of the target processor 50. Therefore if necessary several instructions are required, in order to translate an instruction source program 12. On the other hand can be summarized if necessary several instructions source program 12 by a translated instruction in the goal instruction sequence 18.

The allocation between the respective portion source program 12 and the goal instruction sequence 18 becomes 26 stored in a reference table, see. Double arrow 28. If 12 again executed during the other emulation source program 12 the same portion source program is to become, then become its instructions emulated no longer, but it becomes the associated goal instruction sequence 18 direct on the target processor 50 executed, sees double arrow 30. The emulator program 10 recognizes 26 on the basis an entry in the reference table that for the execution waiting the portion source program 12 already translated is.

Fig. 2 shows a target processor 50 for the treatment of interrupt requests and events. The target processor 50 becomes the execution of the emulator program 10, the translation program 16 and the goal instruction sequences 18 used, sees also Fig. 1. These programmes are in a memory unit 52 stored, on which the processor has 50 over a bus system 54 access.

In an instruction register 56 the current instruction stored which can be implemented by the processor 50 becomes. An instruction counter 58 contains the address of the memory cell in the memory unit 52, is stored in which the instruction which can be worked on. In addition the processor 50 contains register R0 to Rn, become stored in which data words for the instruction execution. The processor 50 contains several functional units, which can work temporal parallel or temporal overlapping on successive instructions. These units are not shown from reasons of the better overview. In addition these functional units are known, z. B. a bus unit, to the access to the bus system 54, a decoding unit for decoding the instruction contained in the instruction register 56, an execution unit for implementing the instruction and a store management unit for converting from memory addresses. For example the microprocessor 80486 contains such functional units of the company Intel.

In a processor act etc. place 60 there are several status bits to the control of the operation of the processor 50. In Fig. 2 is two status bits 62 and 64 shown. The value of the status bits 62 and 64 can be changed by particular register instructions of the command sentence specified for the processor 50. If the status bit 62 has the value ONE, the so processed processor 50 interrupt requests immediately. If the status bit 62 has against it the value ZERO, then interrupt becomes requests in an interruption queue 66, D. h. in a memory unit, to the later processing stored, working after the FIFO principle (roofridge in roofridge Out). The status bit 62 becomes also designated as interrupt flagstone.

The value of the status bit 64 puts fixed, whether events, which affect the state of the processor 50 require however no interruption of the instruction execution, to work on immediately is. If the status bit 64 has the value ONE, then from the outside, over the bus system 54 signaled events can change the state of the processor 50 more immediate. If the status bit 64 has against it the value ZERO, then 54 incoming events become first in an event queue 68 the later evaluation stored over the bus system.

During the emulation achieved becomes by targeted permitting and closing of the processing of interrupt requests that interruptions arise only at predetermined synchronization points. A synchronization point can become inserted, if an instruction source program 12 complete emulated is. For example synchronization points become 12 inserted after the emulation of each fifth instruction source program. Only at the synchronization point the processing of interrupt requests becomes enabled. Between the synchronization points is disabled against it the processing of the interrupt requests. Therefore also if necessary, z can. B. for optimization purposes, the sequence of the emulation by instructions for the origin processor between two synchronization points to be exchanged, as far as no responses between the instructions stand in paths. A change of the status bit 62 for releasing and closing the treatment of interruptions became each time the parallel and/or. overlapped execution in the functional units specified above interrupt. Therefore this bit becomes on the value ZERO set, so that no interruptions become fundamental processed. By particular on the basis the Fig. 4 and 5 explained instructions of the command sentence however nevertheless a treatment of interruptions possible can become.

Likewise it is 12 required with the emulation source program to work on the events only at the synchronization points. Since a change of the status bit 64 has likewise an interruption of the parallel execution in the functional units mentioned to the sequence, this status bit becomes on the value ZERO set, so that no processing of events is fundamental possible. However those becomes down on the basis the Fig by particular instructions. 7 and 8 explained become, the processing of events in the processor 50 nevertheless possible.

Fig. 3 shows method steps if interrupt requests not processed become. The method begins in a method step 100. The method step 100 becomes executed whenever an interrupt request at the processor 50 arrives.

In the method step 104 the arrived interrupt request in the interruption queue becomes 66 stored, sees Fig. 2. Subsequent one becomes maintained on the arrival other interrupt requests. Thus the interrupt requests first only stored ones are not further processed and.

Fig. method steps with the execution of an instruction stored in the instruction register 56 also Fig shows, sees 4. 2. The method begins in a method step 150. In a subsequent method step 152 by the decoding unit the instruction stored in the instruction register 56 is read and decoded.

The processor examines 154 in a next method step whether the status bit 62 has the value ONE. If this is not the case, immediate follows after the method step 154 a method step 156. In the method step 156 tested becomes whether the current instruction stored in the instruction register 56 is an instruction, which releases a test unit, which becomes down 158 explained as method step. Such an instruction would know for example hot "interrupt request examining". In the practice however usually short instruction names become prefered. If the current processed instruction permits interruptions for the instruction cycle required to the processing of this instruction, then immediate follows after the method step 156 a method step 158.

The method step 158 would become immediate 154, if the status bit 62 has the value ONE, D executed after the method step. h. if the processing of interrupt requests is general permissible. In the method step the processor examines 158 whether the interruption queue contains 66 interrupt messages. If an interrupt message is in the interruption queue 66, then immediate follows after the method step 158 a method step 160. In the method step 160 started becomes with the processing of the interrupt request. With the treatment the usual method steps become performed, D. h. Vintages of the interrupt vector and starting a programme to the processing of the interrupt dependent of the interrupt vector. These steps are 162 indicated by a broken arrow.

If 156 found become in the method step that the current instruction which can be implemented is not the instruction "interrupt request examining", then immediate follows after the method step 156 a method step 164, in which the instruction executed stored in the instruction register 56 becomes. Subsequent one becomes the method in the method step 152 continued.

The method step 164 follows also immediate after the method step 158, if in this method step found becomes that in the interrupt memory 66 no interrupt messages is contained.

The execution of the method step 156 requires still no interruption of the parallel instruction execution. Only if 158 found in the method step it becomes that the interrupt memory contains an interrupt message, will the parallel instruction execution interrupted, in order to then begin with the interrupt handling in the method step 160.

Fig. an instruction word 200 for a processor with VLIW architecture (Very Long Instruction Word) shows 5. The instruction word 200 contains three instructions 202 to 206, which form a group, for which it a common information field 208 in the instruction word 200 gives. The information field 208 contains several bit positions, from those in Fig. 5 a bit position 210 highlighted is. If the bit at the bit position 210 has the value ONE, then the test unit becomes 158 performed and an interruption arisen if necessary becomes before the execution of the instructions 202 to 206 processed. Against it if the bit position 210 has the value ZERO, then a processing of interrupt requests is with the execution of the instructions 202 to 206 not possible. The test unit 158 does not become executed. All instruction words processed by the processor contain the information field 208 and therefore also a bit position

▲ top 210 to the control of the treatment of interruptions.

Fig. 6 shows method steps if events not processed become. The method begins in a method step 220, if an event message becomes 54 signaled over the bus system. In a subsequent method step 224 the event message is registered into the event queue 68, since the status bit 64 has the value ZERO. Subsequent one becomes the entry of new events maintained.

The events become the processor 50 for example over instruction words of the command sentence reported. With another embodiment the events over signal lines become 50 signaled to the processor.

Fig. the treatment of events of respective method steps shows 7 with the execution of an instruction by the processor 50. The method begins in a method step 250. In a subsequent method step the instruction standing in the instruction register 56 is decoded by the decoding unit. Subsequent one examines the processor in a subsequent method step 254 on the basis the status bit 64 whether the processing of events is general allowed. The status bit 64 has the value ZERO, D. h. the processing of events is general undue, then an immediate method step 256 follows after the method step 254.

In the method step 256 tested becomes also during general undue treatment of events whether the instruction contained in the instruction register 56 is an instruction Ereignismeldung_prüfen, which releases a test unit. If 256 found become in the method step that the instruction is to become Ereignismeldung_prüfen executed, then an immediate method step 258 follows to the execution of the test unit after the method step 256.

The method step 258 becomes also immediate after the method step 254 executed, if there found becomes that the status bit 64 has the value ONE, D. h. a treatment of events general permissible is. In the method step 258 50 tested become by the processor whether the event queue contains 68 event messages. This is the case, D. h. the event queue 68 is not emptier, then immediate follows after the method step 258 a method step 260.

In the method step 260 event messages are read from the event queue 68. The event specified in the event message becomes subsequent 50 executed of the processor. The state of the processor becomes 50 changed. For example Lookaside Buffer becomes a certain entry for invalid explained in a so called translation. After the processing of the event again with method step 258, sees arrow 262 is continued, until all events are in the event queue 68 processed. On the basis the Fig. 9 down an application example for an event treatment in a multiprocessor system explained becomes.

Against it if 256 found become in the method step that the current instruction which can be implemented is not the instruction event message examining, then immediate follows after the method step 256 a method step 264. In the method step 264 the instruction executed indicated in the instruction register 56 becomes. Subsequent one becomes the method in the method step 252 with the processing of the next instruction continued.

The method step 264 becomes also immediate after the method step 258 executed, if there found becomes that the event queue 68 does not contain event messages.

Fig. an instruction word 300 for a processor shows 8 in accordance with an other embodiment. The instruction word 300 serves for the control of a processor with VLIW architecture (Very Long Instruction Word). The instruction word 300 contains three different instructions 302 to 306, which form a group of instructions. For each group of instructions there is an information field 308 with several bit positions, of those in Fig. 8 a bit position 310 highlighted is. If the bit position 310 has the value ONE, then can become also during general closed treatment of events immediate before the execution of the instructions 302 to 306 dependent of the result of the test unit a treatment of events executed, if event messages are in the event memory 68 stored. Against it if the bit position 310 has the value ZERO, then no test unit becomes executed, if the treatment of events is general disabled.

Each group of instructions which can be worked on from the processor to contains one the bit position 310 of corresponding bit position, so that dependent of each instruction becomes the processing of events allowed or disabled.

Fig. 9 shows the use two processors 400 and 402 for the processing of events with the execution of a store management program 404. The processors 400 and 402 are like above on the basis the Fig. 2 explained processor 50 constructed. The processor 400 contains in particular an address translation table TLB1, in which it is noted whether certain pages of a virtual storage area are in a main memory 406 or in a disk memory 408. The address translation table TLB1 becomes also as translation Lookaside Buffer designated. In the processor 402 there is an address translation table TLB2 with the same function as the address translation table TLB1.

The store management program 404 puts 400 and 402 fixed, which memory sides from the disk memory are 408 into the main memory to transfered during the execution of programmes by the processors. In addition becomes reverse fixed, which to pages of the main memory 406 into the disk memory 408 transfered to become to be supposed, see arrows 410 to 414.

The store management program 404 becomes relocating side entries either on the processor 400 or 402 executed on the processor. 400 and 402 programmes executed on the processors, which emulate source program, become intermediate, which original for the execution by several origin processors programmed is. The origin processors have thereby another type than the processors 400 and 402. To the procedures with the emulation applies in principle when describing the Fig. 1 saying. Additional one must become the operation of the processors 400 and 402 one on the other tuned. For example if a page from the main memory 406 must be written back into the disk memory with implementing the store management program 404 by the processor 400, then a processor-spreading instruction 416 by the processor 400 is to be implemented, by which also the address translation table becomes TLB1 updated. Over bus minutes an event message becomes 419 sent of the processor 400 the processor 402. Additional one must become however likewise ensured that the change of the address translation table TLB1 and TLB2 becomes only 12 performed at certain synchronization points with the emulation source program. In addition test units in the processors know 400 and 402 like above on the basis the processor 50 and the Fig. 4 explained targeted with the help of the instruction event message examining triggered becomes. Arrived events become then processed, see arrows 420 and 422. The event for changing the address translation table TLB1, released by the processor-spreading instruction 416 in the processor 400, is a synchronous event for the processor 400. The event message 419 for changing the address translation table TLB2 is for the processor 402 against it an asynchronous event.

The instruction interruption for Cycle permissible becomes inserted by the emulator program 10 automatic for example after each fifth emulated instruction. Also into the goal instruction sequence 18 16 inserted become in predetermined distances by the translation program.

As on the basis the Fig. 9 to recognize, is suitable the processors 400 and 402 excellent for the emulation of a multiprocessor program. With relatively small effort can be co-ordinated the operation of the processors 400 and 402.

▲ top

Europäisches Patentamt
European Patent Office
Office européen des brevets

Claims of DE10008008	Print	Copy	Contact Us	Close

Result Page

1. Processor (50) for the processing of interrupt requests,
with functional units, the temporal parallel and/or. temporal overlapping at various instructions or at various parts of an instruction works,
and with an interruption processing unit (66), which permits the processing of interrupt requests, whose processing requires an interruption of the work of the functional units in a first mode, and which in a second mode the processing closes from interrupt requests,
characterised in that in the second mode tested becomes whether an interrupt request is to be worked on,
and that dependent of the test result becomes connected when being present an interrupt request into the first mode.

2. Processor (50) according to claim 1, characterised in that the interrupt requests by processor-external procedures generated become.

3. Processor (50) according to claim 1 or 2, characterized by a memory unit (66) to memories of the interrupt requests in the second mode.

4. Processor (50) for the treatment of events, with at least one control unit to the execution of instructions of a command sentence,
characterized by an event processing unit (68), which in a first mode the processing of events permits, by which the switching state of the processor (50) changed, without the work of the control unit becomes interrupted, and which closes the processing of the events in a second mode, whereby the control unit several functional units contains, which work parallel and/or temporal overlapping at various instructions of the command sentence or at various parts of an instruction and that in the second mode tested becomes, whether an event is to be worked on,
and that dependent of the test result becomes connected when being present an event into the first mode.

5. Processor (50) according to claim 4, characterised in that the mode by changing a status bit (64) in a status word (60) to the control of the operation of the processor (50) is more selectable.

6. Processor (50) after one of the claims 4 to 5, characterised in that the events by processor-external procedures (416, 418) caused become.

7. Processor (50) after one of the claims 4 to 6, characterized by a memory unit (68) to memories of event messages with the occurrence of events in the second mode.

8. Processor (50) after one of the preceding claims, characterised in that the test unit by a particular instruction of the command sentence is releasable.

9. Processor (50) according to claim 8, characterised in that the test unit before and/or after the execution of the particular instruction triggered becomes.

10. Processor (50) after one of the claims 1 to 3 and one of the claims 4 to 9 as well as according to claim 10 or 11, characterised in that the particular instruction both the test unit in the interruption processing unit and the test unit in the event processing unit releases.

11. Processor (50) after one of the claims 1 to 7, characterised in that the test unit by an indicator (210; 310) in at least an instruction word (200; 300) triggered becomes.

12. Processor (50) according to claim 11, characterised in that with set indicator (210; 310) the test unit before and/or after the execution of the instruction (202 to 206, 302 to 306) triggered becomes, that by that the indicator (210; 310) contained instruction word (200; 300) fixed is.

13. Processor (50) according to claim 11 or 12, characterised in that the instruction word (200; 300) several instructions (202 to 206, 302 to 306) contains.

14. Processor (50) after one of the claims 1 to 3 and one of the claims 4 to 9 as well as one of the claims 13 to 15, characterised in that the indicator both the test unit in the interruption processing unit and the test unit in the event processing unit releases.

15. Processor (50) after one of the claims 1 to 3 and one of the claims 4 to 9 as well as one of the claims 13 to 15, characterised in that the instruction word for the interruption processing unit and the event processing unit various indicators contains.

16. Processor (50) after one of the preceding claims, characterised in that the processor (50) with the emulation of a processor of other type used becomes.

17. Processor (400, 402) after one of the preceding claims, characterised in that the processor (400, 402) in a multiprocessor system (400, 402) used becomes.

▲ top

(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENT- UND MARKENAMT

(12) **Patentschrift**

(10) **DE 100 08 008 C 1**

(51) Int. Cl.[7]:
G 06 F 9/38
G 06 F 9/40

(21) Aktenzeichen: 100 08 008.1-53
(22) Anmeldetag: 22. 2. 2000
(43) Offenlegungstag: –
(45) Veröffentlichungstag der Patenterteilung: 23. 8. 2001

Innerhalb von 3 Monaten nach Veröffentlichung der Erteilung kann Einspruch erhoben werden

(73) Patentinhaber:
Fujitsu Siemens Computers GmbH, 81739 München, DE

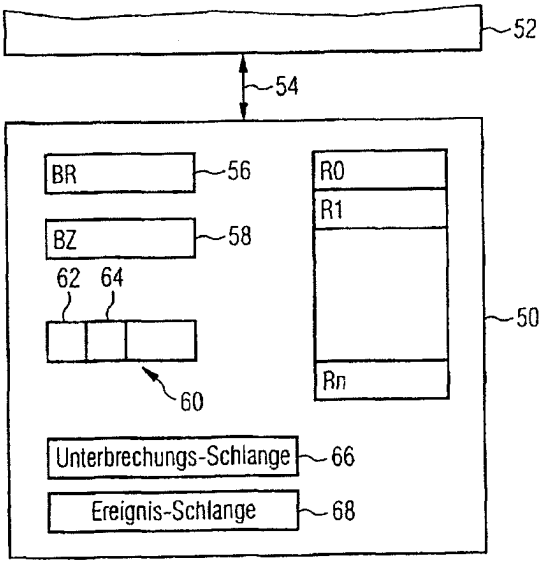
(74) Vertreter:
Epping, W., Dipl.-Ing. Dr.-Ing., Pat.-Anw., 80339 München

(72) Erfinder:
Stadel, Manfred, Dr., 81735 München, DE; Groß, Jürgen, 83026 Rosenheim, DE

(56) Für die Beurteilung der Patentfähigkeit in Betracht gezogene Druckschriften:
EP 05 86 847 A1

(54) Prozessor für die Bearbeitung von Unterbrechungsanforderungen und Prozessor für die Bearbeitung von Ereignissen

(57) Erläutert wird unter anderem ein Prozessor für die Behandlung von Unterbrechungsanforderungen und Ereignissen, der mehrere Funktionseinheiten enthält, die parallel an verschiedenen Befehlen oder an verschiedenen Teilen eines Befehls arbeiten. Der Prozessor enthält außerdem eine Unterbrechungsbearbeitungseinheit, die die Bearbeitung von Unterbrechungsanforderungen bzw. Ereignissen zuläßt oder sperrt. Die Betriebsart "Unterbrechungsanforderung zulassen" wird über einen Prüfschritt ausgewählt.



DE 100 08 008 C 1

DE 100 08 008 C 1

Beschreibung

Die Erfindung betrifft einen Prozessor für die Bearbeitung von Unterbrechungsanforderungen. Der Prozessor enthält Funktionseinheiten, die zeitlich parallel bzw. zeitlich überlappend an verschiedenen Befehlen oder an verschiedenen Teilen eines Befehls arbeiten. Diese Funktionseinheiten werden auch Pipelines genannt. Außerdem enthält der Prozessor eine Unterbrechungsbearbeitungseinheit, die in einer ersten Betriebsart die Bearbeitung von Unterbrechungsanforderungen zuläßt und die in einer zweiten Betriebsart die Bearbeitung von Unterbrechungsanforderungen sperrt. Unterbrechungsanforderungen sind Anforderungen, die eine Unterbrechung der Arbeit der Funktionseinheiten erfordern. Die Unterbrechungsanforderungen werden auch als Interrupts bezeichnet.

Es gibt Unterbrechungsanforderungen, die von außen erzeugt werden, sogenannte asynchrone Unterbrechungen. Außerdem gibt es Unterbrechungsanforderungen, die bei der Ausführung eines Befehls innerhalb des Prozessors erzeugt werden, sogenannte synchrone Unterbrechungen.

Ein Prozessor enthält üblicherweise auf einem Chip ein Rechenwerk und ein Steuerwerk. Bekannte Prozessoren, z. B. der Prozessor Pentium III der Firma Intel, enthalten zum Umschalten der Betriebsart ein Statusbit. Das Statusbit ist Teil eines Statuswortes, das auch als Prozessor-Statuswort oder Maschinen-Statuswort bezeichnet wird. Änderungen im Statuswort führen in der Regel zu einem Abbruch der parallelen bzw. zeitlich überlappenden Verarbeitung der auszuführenden Befehle. Erst nachdem der Prozessor die Statusänderungen bearbeitet hat, wird wieder mit der parallelen bzw. überlappenden Befehlsausführung begonnen.

Aus der EP 0586 847 A1 ist eine Anordnung zur Bearbeitung von Unterbrechungsanforderungen beschrieben. Die Dynamik derartiger Anordnungen hängt weitgehend von der Fähigkeit ab, Systemaufgaben mit sich unterscheidender Dringlichkeit bearbeiten zu können. Die Bearbeitung der Systemaufgaben wird generell durch unterbrechungsgesteuerte Prozesse durchgeführt. Um eine Dynamikverminderung bei der häufigen Verwendung einer Unterbrechungssteuerung für die Bearbeitung einer Kette von Verwaltungsroutinen zu vermeiden, wird bei der beschriebenen Anordnung bei Systemaufgaben bearbeitenden Prozessen, von der unterbrechungsgesteuerten Betriebsart in eine zyklisch abfragegesteuerte Betriebsart gewechselt.

Es ist Aufgabe der Erfindung, einen verbesserten Prozessor für die Behandlung von Unterbrechungsanforderungen anzugeben.

Diese Aufgabe wird durch einen Prozessor mit den Merkmalen des Patentanspruchs 1 gelöst. Weiterbildungen sind in den Unteransprüchen angegeben.

Der Erfindung liegt die Überlegung zugrunde, daß üblicherweise die Bearbeitung von Unterbrechungen während der normalen Ausführung von Programmen zugelassen wird. Jedoch wird während der Ausführung von Programmen zur Behandlung einer Unterbrechungsanforderung die Bearbeitung weiterer Unterbrechungen gesperrt. Die Unterbrechungen werden dabei abhängig von einem Statusbit zugelassen oder gesperrt. Das Statusbit wird im Vergleich zur Anzahl der auszuführenden Befehle sehr selten umgeschaltet, beispielsweise nur alle zehntausend Befehle. Jedoch gibt es Anwendungen, bei denen das Umschalten viel häufiger, z. B. nach der Ausführung von nur fünf bis zehn Befehlen erforderlich wäre. Eine solche Anwendung ist beispielsweise die Emulation eines Ursprungsprozessors bei der Ausführung eines für den Ursprungsprozessor geschriebenen Programms durch einen Zielprozessor, dessen Aufbau sich vom Aufbau des Ursprungsprozessors unterscheidet.

Während der Emulation eines einzigen Befehls des Ursprungsprozessors darf keine Unterbrechung stattfinden, um Inkonsistenz in den während der Emulation nachgebildeten Zuständen des Ursprungsprozessors zu vermeiden. Auf der anderen Seite dürfen auch asynchrone Unterbrechungen nicht zu lange gesperrt sein, wenn durch die Emulation eine ähnliche Bearbeitung von Unterbrechungen wie auf dem Ursprungsprozessor ermöglicht werden soll. Daher muß eine Unterbrechung in nicht allzu großen Abständen, z. B. nach der Emulation von jeweils fünf Befehlen des Ursprungprozessors zugelassen werden. Mit anderen Worten müßte die Betriebsart häufig umgeschaltet werden, um das Bearbeiten von Unterbrechungen zu ermöglichen. Die Unterbrechungen selbst treten jedoch, wie bisher auch, nur relativ selten auf.

Die Erfindung geht weiterhin von der Überlegung aus, daß die Vorteile einer zeitlich parallelen bzw. zeitlich überlappenden Bearbeitung von Befehlen oder von verschiedenen Teilen eines Befehls dann nicht mehr auftreten, wenn die Bearbeitung ständig unterbrochen werden muß. Um diese Vorteile dennoch zu nutzen, wird im erfindungsgemäßen Prozessor in der zweiten Betriebsart geprüft, ob eine Unterbrechungsanforderung zu bearbeiten ist. Abhängig vom Prüfungsergebnis wird nur beim Vorliegen von Unterbrechungsanforderungen in die erste Betriebsart geschaltet.

Der Prüfschritt ermöglicht es, ohne eine vorherige Zustandsänderung des Prozessorstatus gegebenenfalls in die erste Betriebsart umzuschalten. Eine Zustandsänderung des Prozessorstatus würde einen Befehl erfordern, vor dessen Ausführung die Pipelines geleert werden müßten, da erst bei der Befehlsausführung im Prozessor ermittelt wird, welche Statusänderung auftritt und wie schwerwiegend diese ist.

Erst wenn beim Prüfen festgestellt wird, daß Unterbrechungen aufgetreten sind, wird die parallele bzw. überlappende Ausführung von Befehlen beim Umschalten in die erste Betriebsart unterbrochen.

Die Erfindung läßt sich deshalb insbesondere dann einsetzen, wenn ein Wechsel zwischen Sperrung und Zulassung der Bearbeitung von Unterbrechungen häufig erforderlich wäre. Somit gibt es auch andere Anwendungen, als die in dieser Anmeldung genannten Anwendungen. Gerade bei Anwendungen, in denen häufig zwischen der ersten Betriebsart und der zweiten Betriebsart gewechselt werden müßte, wird beim erfindungsgemäßen Prozessor eine erheblich höhere Abarbeitungsgeschwindigkeit der zu diesen Anwendungen gehörenden Programme erreicht als bei bekannten Prozessoren. Die parallele Ausführung von Befehlen wird nämlich nur dann unterbrochen, wenn tatsächlich Unterbrechungsanforderungen zu bearbeiten sind. Das Durchführen des Prüfschritts führt dagegen noch nicht zu einer Unterbrechung der parallelen Befehlsausführung.

Die erfindungsgemäßen Maßnahmen lassen sich an Stelle der bekannten Vorgehensweisen zur Auswahl der Betriebsart oder in Kombination mit diesen Vorgehensweisen einsetzen.

Bei einer Weiterbildung des erfindungsgemäßen Prozessors sind die durch die Unterbrechungsbearbeitungseinheit bearbeiteten Unterbrechungsanforderungen durch prozessorexterne Vorgänge verursacht. Mit anderen Worten sind dies asynchrone Unterbrechungsanforderungen, die beispielsweise durch einen anderen Prozessor gleicher Bauart in einem Mehrprozessorsystem, durch Ein-/Ausgabebausteine oder durch Signale von außen hervorgerufen werden. Diese Ausführungsform ist besonders vorteilhaft, wenn Programme ausgeführt werden sollen, bei denen ohne die Erfindung eine Umschaltung der Betriebsart sehr häufig erforderlich wäre. Eine solche Anwendung ist die Emulation, wie oben bereits ausführlicher erläutert.

Bei einer nächsten Weiterbildung enthält der Prozessor eine Speichereinheit zum Speichern der Unterbrechungsanforderungen in der zweiten Betriebsart. Eine solche Speichereinheit könnte beispielsweise nach dem Warteschlangenprinzip aufgebaut werden, bei dem die in der zweiten Betriebsart auftretenden Unterbrechungsanforderungen in eine Warteschlange eingereiht werden. Erst beim späteren Umschalten in die erste Betriebsart, werden die Unterbrechungsanforderungen der Reihe nach bearbeitet.

Bei einer anderen Ausgestaltung ist der Prozessor so aufgebaut, daß der Prüfschritt durch einen speziellen Befehl des Befehlssatzes ausgelöst wird. Befinden sich bei der Ausführung des speziellen Befehls Unterbrechungsanforderungen in der Warteschlange, so werden diese bearbeitet.

Eine andere Möglichkeit für das Auslösen des Prüfschrittes ist das Verwenden eines Indikators in mindestens einem Befehlswort. Der Indikator ist beispielsweise eine bestimmte Bitstelle. Hat das Bit an dieser Stelle den Wert "1", so wird der Prüfschritt ausgeführt. Beim Wert "0" wird kein Prüfschritt ausgeführt.

Bei einer Ausführungsform wird der Prüfschritt bei gesetztem Indikator vor und nach der Ausführung des Befehls ausgelöst, der den Indikator enthält.

Besonders bei Prozessoren mit VLIW-Architektur (Very Long Instruction Word), bei denen das Befehlswort mehrere Befehle enthält, läßt sich ein Indikator vorteilhaft verwenden, da ein zusätzliches Bit nur für eine Befehlsgruppe eines VLIW erforderlich ist.

Die Erfindung betrifft für die Behandlung von Ereignissen außerdem einen Prozessor, der mindestens eine Steuereinheit zur Ausführung von Befehlen eines Befehlssatzes enthält. Eine Steuereinheit wird benötigt, wenn Befehle nur nacheinander oder zeitlich teilweise überlappend ausgeführt werden sollen. Mehrere Steuereinheiten sind bei der parallelen Ausführung von Befehlen oder Befehlsteilen vorhanden.

Durch ein Ereignis wird der Schaltzustand des Prozessors verändert. Jedoch muß im Unterschied zu einer Unterbrechungsanforderung die Arbeit der Steuereinheit nicht zwingend unterbrochen werden. Es lassen sich auch bei Ereignissen asynchrone und synchrone Ereignisse unterscheiden. Asynchrone Ereignisse sind Ereignisse, die auf Vorgänge außerhalb des Prozessors zurückzuführen sind, beispielsweise auf Vorgänge in anderen Prozessoren desselben Mehrprozessorsystems, z. B. prozessorübergreifende Befehle. Synchrone Ereignisse sind Ereignisse, die bei der Ausführung eines Befehls im Prozessor selbst entstehen.

Die Ereignisse werden bei bekannten Prozessoren behandelt, sobald sie auftreten. Dies führt dazu, daß in einem Mehrprozessorsystem prozessorübergreifende Befehle nur eingeschränkt genutzt werden können. Außerdem sind aufwendige Programme erforderlich, um das Zusammenarbeiten der Prozessoren zu ermöglichen.

Die Schwierigkeiten werden noch größer, wenn Programme eines Ursprungs-Mehrprozessorsystems durch Ziel-Prozessoren eines Ziel-Mehrprozessorsystems ausgeführt werden sollen. Für eine solche Emulation sind die bekannten Prozessoren nur bedingt geeignet.

Es ist Aufgabe eines zweiten Aspekts der Erfindung, einen verbesserten Prozessor für die Behandlung von Ereignissen anzugeben.

Diese Aufgabe wird durch einen Prozessor mit den Merkmalen des Patentanspruchs 4 gelöst. Weiterbildungen sind in den Unteransprüchen angegeben.

Die Erfindung gemäß zweitem Aspekt geht von der Überlegung aus, daß die Hauptursache für die oben genannten Probleme in einer fehlenden Ereignisbearbeitungseinheit zu sehen ist, die eine Möglichkeit schafft, die Bearbeitung von Ereignissen zu verzögern. Deshalb enthält der erfindungsgemäße Prozessor eine Ereignisbearbeitungseinheit, die in einer ersten Betriebsart die Bearbeitung von Ereignissen zuläßt und die in einer zweiten Betriebsart die Bearbeitung der Ereignisse sperrt. In der zweiten Betriebsart der Ereignisbearbeitungseinheit ist gewährleistet, daß der Prozessor ungestört durch das Ereignis Befehle ausführt. Der Wechsel der Betriebsarten kann von außen zu vorgegebenen Zeitpunkten erfolgen. Dadurch läßt sich die Arbeitsweise von Prozessoren in einem Mehrprozessorsystem aufeinander abstimmen.

Bei einer Ausgestaltung enthält die Steuereinheit des Prozessors mehrere Funktionseinheiten, die zeitlich parallel bzw. zeitlich überlappend an verschiedenen Befehlen des Befehlssatzes oder an verschiedenen Teilen eines Befehls arbeiten. Gerade bei Prozessoren, die derartig komplexe Steuereinheiten haben, führt eine Ereignisbearbeitungseinheit beim Einsatz in einem Mehrprozessorsystem zu einer erheblichen Vereinfachung bei der Abstimmung der Betriebsweise der Prozessoren aufeinander.

Die Betriebsart läßt sich beispielsweise durch das Ändern eines Statusbits in einem Statuswort zur Steuerung der Arbeitsweise des Prozessors auswählen. Jedoch hat dies bei Prozessoren, mit mehreren Steuereinheiten zur parallelen Ausführung verschiedener Befehle oder Befehlsteile den Nachteil, daß bei einer Änderung des Statusbits die parallele Ausführung von Befehlen kurzzeitig unterbrochen werden muß. Das Ändern eines Statusbits ist jedoch eine einfache technische Maßnahme, die keinen zusätzlichen Befehl im Befehlssatz erfordert. Für viele Anwendungen ist das Ändern eines Statusbits und die damit verbundene Unterbrechung auch hinnehmbar.

Bei einer alternativen bzw. kummulativen Ausgestaltung wird in der zweiten Betriebsart geprüft, ob ein Ereignis zu bearbeiten ist. Abhängig vom Prüfergebnis wird beim Vorliegen eines Ereignisses in die erste Betriebsart geschaltet. Insbesondere wird beispielsweise kein Statusbit in einem Statuswort geändert, in dem auch Zustände verzeichnet sind, die eine Unterbrechung der Arbeit der Funktionseinheiten erfordern. Der Prozessor müßte dann nämlich vor der Bearbeitung des das Statuswort ändernden Befehls die parallele Bearbeitung unterbrechen, da er erst bei der Bearbeitung dieses Befehls feststellen kann, welches Statusbit geändert worden ist. Stellt sich heraus, daß keine Unterbrechung der Arbeit der Funktionseinheiten erforderlich war, so läßt sich die Unterbrechung nicht mehr rückgängig machen.

Bei einer Weiterbildung sind die Ereignisse durch prozessorexterne Vorgänge hervorgerufen. Mit anderen Worten handelt es sich um asynchrone Ereignisse. Solche Ereignisse sind z. B.:

- die prozessorübergreifende Invalidierung eines sogenannten TL-Buffers (Translation Lookaside Buffer), oder
- Vorgänge im Zusammenhang mit dem Betrieb eines schnellen Zwischenspeichers (Cache).

Bei einer nächsten Ausgestaltung enthält der Prozessor eine Speichereinheit zum Speichern von Ereignismeldungen beim Auftreten von Ereignissen in der zweiten Betriebsart. Die Bearbeitung der Ereignisse kann durch das Speichern in der Speichereinheit bis zum Umschalten in die erste Betriebsart verzögert werden. Die Verzögerungszeit läßt sich durch gezieltes Auslösen des Prüfschritts vorgeben, so daß insbesondere Prozessoren in einem Mehrprozessorsystem synchronisiert werden können.

Der Prüfschritt wird bei einer Ausgestaltung durch einen speziellen Befehl des Befehlssatzes ausgelöst. Dies führt insbesondere bei Programmen, bei denen die Betriebsart sonst häufig umgeschaltet werden müßte, zu erheblich kür-

zeren Programmen. Ein Beispiel für solche Programme sind Emulationsprogramme.

Alternativ wird der Prüfschritt durch einen Indikator in mindestens einem Befehlswort ausgelöst. Bei einer Ausführungsform enthalten alle Befehle des Befehlssatzes diesen Indikator. Der Indikator ist beispielsweise eine bestimmte Bitstelle. Bei VLIW-Prozessoren (Very Long Instruction Word) enthält das Befehlswort mehrere Befehle. Der Indikator betrifft dann mehrere Befehle gleichzeitig.

Enthält der Prozessor sowohl eine Unterbrechungsbearbeitungseinheit als auch eine Ereignisbearbeitungseinheit, so wird bei einer Ausgestaltung mit Hilfe eines Befehlswortes der Prüfschritt sowohl in der Unterbrechungsbearbeitungseinheit als auch in der Ereignisbearbeitungseinheit ausgelöst. So läßt sich beispielsweise die Bearbeitung von asynchronen Unterbrechungsanforderungen und asynchronen Ereignissen mit Hilfe eines Befehls auslösen.

Bei einem Prozessor mit Unterbrechungsbearbeitungseinheit und Ereignisbearbeitungseinheit lassen sich in einem, mehreren oder allen Befehlsworten jeweils zwei verschiedene Indikatoren oder ein gemeinsamer Indikator zum Auslösen der Prüfschritte in der Unterbrechungsbearbeitungseinheit und der Ereignisbearbeitungseinheit verwenden.

Wird der erfindungsgemäße Prozessor für die Behandlung von Ereignissen bzw. eine seiner Ausgestaltungen in einem Mehrprozessorsystem eingesetzt, so lassen sich prozessorübergreifende Befehle verwenden, auch wenn diese auf Synchronisationspunkte in anderen Prozessoren des Mehrprozessorsystems Rücksicht nehmen müssen. Durch eine geeignete Wahl des Auslösens der Prüfschritte in der Ereignisbearbeitungseinheit läßt sich nämlich gewährleisten, daß die Arbeitsweise eines Prozessors nicht durch die anderen Prozessoren gestört wird.

Im folgenden werden die beiden Aspekte der Erfindung an Hand der beiliegenden Zeichnungen erläutert. Darin zeigen:

Fig. 1 Vorgänge beim Ausführen eines Emulationsprogramms,

Fig. 2 einen Prozessor für die Behandlung von Unterbrechungsanforderungen und Ereignissen,

Fig. 3 Verfahrensschritte für den Fall, daß Unterbrechungsanforderungen nicht bearbeitet werden können,

Fig. 4 Verfahrensschritte bei der Ausführung eines Befehls durch den Prozessor,

Fig. 5 ein Befehlswort für einen VLIW-Prozessor,

Fig. 6 Verfahrensschritte für den Fall, daß Ereignisse nicht bearbeitet werden,

Fig. 7 die Ereignisbearbeitung betreffende Verfahrensschritte bei der Ausführung eines Befehls durch den Prozessor,

Fig. 8 ein Befehlswort für einen VLIW-Prozessor, und

Fig. 9 den Einsatz zweier Prozessoren für die Behandlung von Ereignissen bei der Ausführung eines Speicher verwaltungsprogramms.

Fig. 1 zeigt Vorgänge beim Ausführen eines Emulatorprogramms **10**, dessen Befehle durch einen Ziel-Prozessor **50** abgearbeitet werden, dessen Aufbau unten an Hand der **Fig.** 2 näher erläutert wird. Das Emulatorprogramm **10** führt Befehle eines Ursprungsprogramms **12** aus. Die Befehle des Ursprungsprogramms sind durch den Befehlssatz eines Ursprungsprozessors vorgegeben, dessen Aufbau sich vom Befehlssatz des an Hand der **Fig.** 2 erläuterten Ziel-Prozessors **50** unterscheidet. Die Befehle des Ursprungsprogramms **12** werden vom Emulatorprogramm **10** nacheinander bearbeitet, siehe Pfeil **14**. Zunächst wird das Bitmuster jedes Befehlswortes in seine funktionalen Bestandteile zerlegt, d. h. beispielsweise in den Operationscode, in Registeroperanden und Direktoperanden. Anschließend lädt das Emulatorprogramm **10** die Werte der Operanden und führt die durch den Operationscode bestimmte Operation aus. Das Ergebnis wird für die weitere Bearbeitung z. B. in Arbeitsregistern des Ziel-Prozessors gespeichert. Danach wird der im Emulatorprogramm **10** nachgebildete Programmzähler des Ursprungsprozessors inkrementiert.

Bei der Emulation des Ursprungsprogramms **12** registriert das Emulatorprogramm **10** die Häufigkeit der Ausführung bestimmter Abschnitte des Ursprungsprogramms **12**. Wird ein Programmstück sehr oft ausgeführt, so startet das Emulatorprogramm **10** ein Übersetzungsprogramm **16**. Beim Ausführen des Übersetzungsprogramms **16** wird der betreffende Abschnitt des Ursprungsprogramms **12** in eine Zielbefehlsfolge **18** übersetzt, die Befehle des Befehlssatzes für den Ziel-Prozessor enthält, siehe Pfeile **20** bis **24**. Es gibt nicht für jeden Befehl des Ursprungsprogramms **12** einen entsprechenden Befehl im Befehlssatz des Ziel-Prozessors **50**. Deshalb sind gegebenenfalls mehrere Befehle erforderlich, um einen Befehl des Ursprungsprogramms **12** zu übersetzen. Andererseits lassen sich gegebenenfalls mehrere Befehle des Ursprungsprogramms **12** durch einen übersetzten Befehl in der Zielbefehlsfolge **18** zusammenfassen.

Die Zuordnung zwischen dem jeweiligen Abschnitt des Ursprungsprogramms **12** und der Zielbefehlsfolge **18** wird in einer Adreßzuordnungstabelle **26** gespeichert, vgl. Doppelpfeil **28**. Wenn bei der weiteren Emulation des Ursprungsprogramms **12** derselbe Abschnitt des Ursprungsprogramms **12** erneut ausgeführt werden soll, so werden seine Befehle nicht mehr emuliert, sondern es wird die zugehörige Zielbefehlsfolge **18** direkt auf dem Ziel-Prozessor **50** ausgeführt, siehe Doppelpfeil **30**. Das Emulatorprogramm **10** erkennt an Hand eines Eintrags in der Adreßzuordnungstabelle **26**, daß der zur Ausführung anstehende Abschnitt des Ursprungsprogramms **12** bereits übersetzt worden ist.

Fig. 2 zeigt einen Ziel-Prozessor **50** für die Behandlung von Unterbrechungsanforderungen und Ereignissen. Der Ziel-Prozessor **50** wird zur Ausführung des Emulatorprogramms **10**, des Übersetzungsprogramms **16** und der Zielbefehlsfolgen **18** eingesetzt, siehe auch **Fig.** 1. Diese Programme sind in einer Speichereinheit **52** gespeichert, auf die der Prozessor **50** über ein Bussystem **54** Zugriff hat.

In einem Befehlsregister **56** wird der momentan durch den Prozessor **50** auszuführende Befehl gespeichert. Ein Befehlszähler **58** enthält die Adresse der Speicherzelle in der Speichereinheit **52**, in der der zu bearbeitende Befehl gespeichert ist. Der Prozessor **50** enthält außerdem Register R0 bis Rn, in denen Datenworte für die Befehlsausführung gespeichert werden. Der Prozessor **50** enthält mehrere Funktionseinheiten, die zeitlich parallel oder zeitlich überlappend an aufeinanderfolgenden Befehlen arbeiten können. Diese Einheiten sind aus Gründen der besseren Übersicht nicht dargestellt. Außerdem sind diese Funktionseinheiten bekannt, z. B. eine Buseinheit, zum Zugriff auf das Bussystem **54**, eine Decodiereinheit zum Decodieren des im Befehlsregister **56** enthaltenen Befehls, eine Ausführungseinheit zum Ausführen des Befehls und eine Speicherverwaltungseinheit zum Umwandeln von Speicheradressen. Beispielsweise enthält der Mikroprozessor 80486 der Firma Intel solche Funktionseinheiten.

In einem Prozessorstatuswort **60** gibt es mehrere Statusbits zur Steuerung der Arbeitsweise des Prozessors **50**. In **Fig.** 2 sind zwei Statusbits **62** und **64** dargestellt. Der Wert der Statusbits **62** und **64** läßt sich mit Hilfe von speziellen Registerbefehlen des für den Prozessor **50** festgelegten Befehlssatzes ändern. Hat das Statusbit **62** den Wert EINS, so bearbeitet der Prozessor **50** Interrupt-Anforderungen sogleich. Hat das Statusbit **62** dagegen den Wert NULL, so werden Interrupt-Anforderungen in einer Unterbrechungs-

Schlange **66**, d. h. in einer nach dem FIFO-Prinzip (First In First Out) arbeitenden Speichereinheit, zur späteren Bearbeitung gespeichert. Das Statusbit **62** wird auch als Interrupt-Flag bezeichnet.

Der Wert des Statusbits **64** legt fest, ob Ereignisse, die den Zustand des Prozessors **50** beeinflussen, jedoch keine Unterbrechung der Befehlsausführung erfordern, sogleich zu bearbeiten sind. Hat das Statusbit **64** den Wert EINS, so können von außen, über das Bussystem **54** signalisierte Ereignisse den Zustand des Prozessors **50** unmittelbar verändern. Hat das Statusbit **64** dagegen den Wert NULL, so werden über das Bussystem **54** eintreffende Ereignisse zunächst in einer Ereignis-Schlange **68** zur späteren Auswertung gespeichert.

Bei der Emulation wird durch gezieltes Zulassen und Sperren der Bearbeitung von Unterbrechungsanforderungen erreicht, daß Unterbrechungen nur an vorgegebenen Synchronisationspunkten auftreten. Ein Synchronisationspunkt kann eingefügt werden, wenn ein Befehl des Ursprungsprogramms **12** vollständig emuliert worden ist. Beispielsweise werden Synchronisationspunkte nach der Emulation jedes fünften Befehls des Ursprungsprogramms **12** eingefügt. Erst am Synchronisationspunkt wird die Bearbeitung von Unterbrechungsanforderungen freigegeben. Zwischen den Synchronisationspunkten ist dagegen die Bearbeitung der Unterbrechungsanforderungen gesperrt. Deshalb kann auch bei Bedarf, z. B. zu Optimierungszwecken, die Reihenfolge der Emulation von Befehlen für den Ursprungsprozessor zwischen zwei Synchronisationspunkten vertauscht werden, soweit keine Abhängigkeiten zwischen den Befehlen im Wege stehen. Eine Veränderung des Statusbits **62** zum Freigeben und Sperren der Unterbrechungsbearbeitung würde jedesmal die parallele bzw. überlappte Ausführung in den oben genannten Funktionseinheiten unterbrechen. Deshalb wird dieses Bit auf den Wert NULL gesetzt, so daß grundsätzlich keine Unterbrechungen bearbeitet werden. Durch spezielle an Hand der **Fig.** 4 und 5 erläuterte Befehle des Befehlssatzes kann jedoch trotzdem eine Unterbrechungsbearbeitung ermöglicht werden.

Ebenso ist es bei der Emulation des Ursprungsprogramms **12** erforderlich, die Ereignisse nur an den Synchronisationspunkten zu bearbeiten. Da eine Veränderung des Statusbits **64** ebenfalls eine Unterbrechung der parallelen Ausführung in den genannten Funktionseinheiten zur Folge hat, wird dieses Statusbit auf den Wert NULL gesetzt, so daß grundsätzlich keine Bearbeitung von Ereignissen möglich ist. Jedoch wird mit Hilfe von speziellen Befehlen, die unten an Hand der **Fig.** 7 und 8 erläutert werden, die Bearbeitung von Ereignissen im Prozessor **50** trotzdem ermöglicht.

Fig. 3 zeigt Verfahrensschritte für den Fall, daß Unterbrechungsanforderungen nicht bearbeitet werden. Das Verfahren beginnt in einem Verfahrensschritt **100**. Der Verfahrensschritt **100** wird immer dann ausgeführt, wenn eine Unterbrechungsanforderung am Prozessor **50** eintrifft.

Im Verfahrensschritt **104** wird die eingetroffene Unterbrechungsanforderung in der Unterbrechungs-Schlange **66** gespeichert, siehe **Fig.** 2. Anschließend wird auf das Eintreffen weiterer Unterbrechungsanforderungen gewartet. Somit werden die Unterbrechungsanforderungen zunächst nur gespeichert und nicht weiterbearbeitet.

Fig. 4 zeigt Verfahrensschritte bei der Ausführung eines im Befehlsregister **56** gespeicherten Befehls, siehe auch **Fig.** 2. Das Verfahren beginnt in einem Verfahrensschritt **150**. In einem folgenden Verfahrensschritt **152** wird durch die Decodierungseinheit der im Befehlsregister **56** gespeicherte Befehl gelesen und decodiert.

Der Prozessor prüft in einem nächsten Verfahrensschritt **154**, ob das Statusbit **62** den Wert EINS hat. Wenn dies nicht der Fall ist, folgt unmittelbar nach dem Verfahrensschritt **154** ein Verfahrensschritt **156**. Im Verfahrensschritt **156** wird geprüft, ob der momentan im Befehlsregister **56** gespeicherte Befehl ein Befehl ist, der einen Prüfschritt auslöst, der unten als Verfahrensschritt **158** erläutert wird. Ein solcher Befehl könnte beispielsweise heißen "Unterbrechungsanforderungprüfen". In der Praxis werden jedoch meist kurze Befehlsnamen bevorzugt. Läßt der aktuell bearbeitet Befehl Unterbrechungen für den zur Bearbeitung dieses Befehls erforderlichen Befehlszyklus zu, so folgt unmittelbar nach dem Verfahrensschritt **156** ein Verfahrensschritt **158**.

Der Verfahrensschritt **158** würde unmittelbar nach dem Verfahrensschritt **154** ausgeführt werden, wenn das Statusbit **62** den Wert EINS hat, d. h. wenn die Bearbeitung von Unterbrechungsanforderungen generell zulässig ist. Im Verfahrensschritt **158** prüft der Prozessor, ob die Unterbrechungs-Schlange **66** Interrupt-Meldungen enthält. Befindet sich eine Interrupt-Meldung in der Unterbrechungs-Schlange **66**, so folgt unmittelbar nach dem Verfahrensschritt **158** ein Verfahrensschritt **160**. Im Verfahrensschritt **160** wird mit der Bearbeitung der Unterbrechungsanforderung begonnen. Bei der Bearbeitung werden die üblichen Verfahrensschritte durchgeführt, d. h. Lesen des Interrupt-Vektors und Starten eines Programms zur Bearbeitung des Interrupts abhängig vom Interrupt-Vektor. Diese Schritte sind durch einen gestrichelten Pfeil **162** angedeutet.

Wird im Verfahrensschritt **156** festgestellt, daß der aktuell auszuführende Befehl nicht der Befehl "Unterbrechungsanforderungprüfen" ist, so folgt unmittelbar nach dem Verfahrensschritt **156** ein Verfahrensschritt **164**, in dem der im Befehlsregister **56** gespeicherte Befehl ausgeführt wird. Anschließend wird das Verfahren im Verfahrensschritt **152** fortgesetzt.

Der Verfahrensschritt **164** folgt auch unmittelbar nach dem Verfahrensschritt **158**, wenn in diesem Verfahrensschritt festgestellt wird, daß im Interrupt-Speicher **66** keine Interrupt-Meldungen enthalten sind.

Die Ausführung des Verfahrensschrittes **156** erfordert noch keine Unterbrechung der parallelen Befehlsausführung. Erst wenn im Verfahrensschritt **158** festgestellt wird, daß der Interrupt-Speicher eine Interrupt-Meldung enthält, wird die parallele Befehlsausführung unterbrochen, um dann mit der Unterbrechungsbehandlung im Verfahrensschritt **160** zu beginnen.

Fig. 5 zeigt ein Befehlswort **200** für einen Prozessor mit VLIW-Architektur (Very Long Instruction Word). Das Befehlswort **200** enthält drei Befehle **202** bis **206**, die eine Gruppe bilden, für die es ein gemeinsames Informationsfeld **208** im Befehlswort **200** gibt. Das Informationsfeld **208** enthält mehrere Bitstellen, von denen in **Fig.** 5 eine Bitstelle **210** hervorgehoben ist. Hat das Bit an der Bitstelle **210** den Wert EINS, so wird der Prüfschritt **158** durchgeführt und eine gegebenenfalls aufgetretene Unterbrechung wird vor der Ausführung der Befehle **202** bis **206** bearbeitet. Hat dagegen die Bitstelle **210** den Wert NULL, so ist eine Bearbeitung von Unterbrechungsanforderungen bei der Ausführung der Befehle **202** bis **206** nicht möglich. Der Prüfschritt **158** wird nicht ausgeführt. Alle durch den Prozessor bearbeiteten Befehlsworte enthalten das Informationsfeld **208** und demzufolge auch eine Bitstelle **210** zur Steuerung der Unterbrechungsbearbeitung.

Fig. 6 zeigt Verfahrensschritte für den Fall, daß Ereignisse nicht bearbeitet werden. Das Verfahren beginnt in einem Verfahrensschritt **220**, wenn eine Ereignismeldung über das Bussystem **54** signalisiert wird. In einem folgenden Verfahrensschritt **224** wird die Ereignismeldung in die Ereignis-Schlange **68** eingetragen, da das Statusbit **64** den

Wert NULL hat. Anschließend wird wieder auf den Eintritt neuer Ereignisse gewartet.

Die Ereignisse werden dem Prozessor **50** beispielsweise über Befehlsworte des Befehlssatzes mitgeteilt. Bei einem anderen Ausführungsbeispiel werden die Ereignisse über Signalleitungen an den Prozessor **50** signalisiert.

Fig. 7 zeigt die Ereignisbearbeitung betreffende Verfahrensschritte bei der Ausführung eines Befehls durch den Prozessor **50**. Das Verfahren beginnt in einem Verfahrensschritt **250**. In einem folgenden Verfahrensschritt wird der im Befehlsregister **56** stehende Befehl durch die Decodiereinheit decodiert. Anschließend prüft der Prozessor in einem folgenden Verfahrensschritt **254** an Hand des Statusbits **64**, ob die Bearbeitung von Ereignissen generell zugelassen ist. Hat das Statusbit **64** den Wert NULL, d. h. die Bearbeitung von Ereignissen ist generell unzulässig, so folgt nach dem Verfahrensschritt **254** unmittelbar ein Verfahrensschritt **256**.

Im Verfahrensschritt **256** wird auch bei generell unzulässiger Ereignisbearbeitung geprüft, ob der im Befehlsregister **56** enthaltene Befehl ein Befehl Ereignismeldung_prüfen ist, der einen Prüfschritt auslöst. Wird im Verfahrensschritt **256** festgestellt, daß der Befehl Ereignismeldung_prüfen ausgeführt werden soll, so folgt nach dem Verfahrensschritt **256** unmittelbar ein Verfahrensschritt **258** zur Ausführung des Prüfschritts.

Der Verfahrensschritt **258** wird auch unmittelbar nach dem Verfahrensschritt **254** ausgeführt, wenn dort festgestellt wird, daß das Statusbit **64** den Wert EINS hat, d. h. eine Ereignisbearbeitung generell zulässig ist. Im Verfahrensschritt **258** wird durch den Prozessor **50** geprüft, ob die Ereignis-Schlange **68** Ereignismeldungen enthält. Ist dies der Fall, d. h. die Ereignis-Schlange **68** ist nicht leer, so folgt unmittelbar nach dem Verfahrensschritt **258** ein Verfahrensschritt **260**.

Im Verfahrensschritt **260** werden Ereignismeldungen aus der Ereignis-Schlange **68** gelesen. Das in der Ereignismeldung spezifizierte Ereignis wird anschließend vom Prozessor **50** ausgeführt. Dabei wird der Zustand des Prozessors **50** verändert. Beispielsweise wird in einem sogenannten Translation-Lookaside-Buffer ein bestimmter Eintrag für ungültig erklärt. Nach der Bearbeitung des Ereignisses wird wiederum mit Verfahrensschritt **258** fortgefahren, siehe Pfeil **262**, bis alle Ereignisse in der Ereignis-Schlange **68** bearbeitet sind. An Hand der **Fig. 9** wird unten ein Anwendungsbeispiel für eine Ereignisbehandlung in einem Mehrprozessorsystem erläutert.

Wird dagegen im Verfahrensschritt **256** festgestellt, daß der aktuell auszuführende Befehl nicht der Befehl Ereignismeldungprüfen ist, so folgt unmittelbar nach dem Verfahrensschritt **256** ein Verfahrensschritt **264**. Im Verfahrensschritt **264** wird der im Befehlsregister **56** angegebene Befehl ausgeführt. Anschließend wird das Verfahren im Verfahrensschritt **252** mit der Bearbeitung des nächsten Befehls fortgesetzt.

Der Verfahrensschritt **264** wird auch unmittelbar nach dem Verfahrensschritt **258** ausgeführt, wenn dort festgestellt wird, daß die Ereignis-Schlange **68** keine Ereignismeldungen enthält.

Fig. 8 zeigt ein Befehlswort **300** für einen Prozessor gemäß einem weiteren Ausführungsbeispiel. Das Befehlswort **300** dient zur Steuerung eines Prozessors mit VLIW-Architektur (Very Long Instruction Word). Das Befehlswort **300** enthält drei unterschiedliche Befehle **302** bis **306**, die eine Befehlsgruppe bilden. Für jede Befehlsgruppe gibt es ein Informationsfeld **308** mit mehreren Bitstellen, von denen in **Fig. 8** eine Bitstelle **310** hervorgehoben ist. Hat die Bitstelle **310** den Wert EINS, so kann auch bei generell gesperrter Ereignisbearbeitung unmittelbar vor der Ausführung der Befehle **302** bis **306** abhängig vom Ergebnis des Prüfschritts eine Ereignisbearbeitung ausgeführt werden, falls Ereignismeldungen im Ereignisspeicher **68** gespeichert sind. Hat dagegen die Bitstelle **310** den Wert NULL, so wird kein Prüfschritt ausgeführt, wenn die Ereignisbearbeitung generell gesperrt ist.

Jede vom Prozessor zu bearbeitende Befehlsgruppe enthält eine der Bitstelle **310** entsprechende Bitstelle, so daß abhängig von jedem Befehl die Bearbeitung von Ereignissen zugelassen oder gesperrt wird.

Fig. 9 zeigt den Einsatz zweier Prozessoren **400** und **402** für die Bearbeitung von Ereignissen bei der Ausführung eines Speicherverwaltungsprogramms **404**. Die Prozessoren **400** und **402** sind wie der oben an Hand der **Fig. 2** erläuterte Prozessor **50** aufgebaut. Der Prozessor **400** enthält insbesondere eine Adreßübersetzungstabelle TLB1, in der vermerkt wird, ob sich bestimmte Seiten eines virtuellen Speicherbereichs in einem Hauptspeicher **406** oder in einem Plattenspeicher **408** befinden. Die Adreßübersetzungstabelle TLB1 wird auch als Translation-Lookaside-Buffer bezeichnet. Im Prozessor **402** gibt es eine Adreßübersetzungstabelle TLB2 mit der gleichen Funktion wie die Adreßübersetzungstabelle TLB1.

Das Speicherverwaltungsprogramm **404** legt während der Ausführung von Programmen durch die Prozessoren **400** und **402** fest, welche Speicherseiten aus dem Plattenspeicher **408** in den Hauptspeicher zu übertragen sind. Außerdem wird umgekehrt festgelegt, welche Seiten vom Hauptspeicher **406** in den Plattenspeicher **408** übertragen werden sollen, siehe Pfeile **410** bis **414**.

Das Speicherverwaltungsprogramm **404** wird zum Umspeichern von Seiteneinträgen entweder auf dem Prozessor **400** oder auf dem Prozessor **402** ausgeführt. Zwischenzeitlich werden auf den Prozessoren **400** und **402** Programme ausgeführt, die ein Ursprungsprogramm emulieren, das ursprünglich für die Ausführung durch mehrere Ursprungsprozessoren programmiert worden ist. Die Ursprungsprozessoren haben dabei eine andere Bauart als die Prozessoren **400** und **402**. Für die Vorgänge bei der Emulation gilt grundsätzlich das beim Erläutern der **Fig. 1** Gesagte. Zusätzlich muß die Arbeitsweise der Prozessoren **400** und **402** aufeinander abgestimmt werden. Muß beim Ausführen des Speicherverwaltungsprogramms **404** durch den Prozessor **400** beispielsweise eine Seite aus dem Hauptspeicher **406** in den Plattenspeicher zurückgeschrieben werden, so ist ein prozessorübergreifender Befehl **416** vom Prozessor **400** auszuführen, durch den auch die Adreßübersetzungstabelle TLB1 aktualisiert wird. Über ein Busprotokoll wird vom Prozessor **400** zum Prozessor **402** eine Ereignismeldung **419** gesendet. Zusätzlich muß jedoch ebenfalls gewährleistet werden, daß die Veränderung der Adreßübersetzungstabelle TLB1 und TLB2 nur an bestimmten Synchronisationspunkten bei der Emulation des Ursprungsprogramms **12** durchgeführt wird. Dazu können Prüfschritte in den Prozessoren **400** und **402** wie oben an Hand des Prozessors **50** und der **Fig. 4** erläutert gezielt mit Hilfe des Befehls Ereignismeldungprüfen ausgelöst werden. Eingetroffene Ereignisse werden dann bearbeitet, siehe Pfeile **420** und **422**. Das durch den prozessorübergreifenden Befehl **416** im Prozessor **400** ausgelöste Ereignis zum Ändern der Adreßübersetzungstabelle TLB1 ist ein synchrones Ereignis für den Prozessor **400**. Die Ereignismeldung **419** zum Ändern der Adreßübersetzungstabelle TLB2 ist für den Prozessor **402** dagegen ein asynchrones Ereignis.

Der Befehl Unterbrechung für Zyklus zulässig wird durch das Emulatorprogramm **10** automatisch beispielsweise nach jedem fünften emulierten Befehl eingefügt. Auch in die

Zielbefehlsfolge **18** werden in vorgegebenen Abständen durch das Übersetzungsprogramm **16** eingefügt.

Wie an Hand der **Fig. 9** zu erkennen, eignen sich die Prozessoren **400** und **402** hervorragend für die Emulation eines Mehrprozessorprogramms. Mit verhältnismäßig geringem Aufwand läßt sich die Arbeitsweise der Prozessoren **400** und **402** aufeinander abstimmen.

Patentansprüche

1. Prozessor (**50**) für die Bearbeitung von Unterbrechungsanforderungen,
mit Funktionseinheiten, die zeitlich parallel bzw. zeitlich überlappend an verschiedenen Befehlen oder an verschiedenen Teilen eines Befehls arbeiten,
und mit einer Unterbrechungsbearbeitungseinheit (**66**), die in einer ersten Betriebsart die Bearbeitung von Unterbrechungsanforderungen zuläßt, deren Bearbeitung eine Unterbrechung der Arbeit der Funktionseinheiten erfordert, und die in einer zweiten Betriebsart die Bearbeitung von Unterbrechungsanforderungen sperrt,
dadurch gekennzeichnet, daß in der zweiten Betriebsart geprüft wird, ob eine Unterbrechungsanforderung zu bearbeiten ist,
und daß abhängig vom Prüfergebnis beim Vorliegen einer Unterbrechungsanforderung in die erste Betriebsart geschaltet wird.

2. Prozessor (**50**) nach Anspruch 1, dadurch gekennzeichnet, daß die Unterbrechungsanforderungen durch prozessorexterne Vorgänge erzeugt werden.

3. Prozessor (**50**) nach Anspruch 1 oder 2, gekennzeichnet durch eine Speichereinheit (**66**) zum Speichern der Unterbrechungsanforderungen in der zweiten Betriebsart.

4. Prozessor (**50**) für die Behandlung von Ereignissen, mit mindestens einer Steuereinheit zur Ausführung von Befehlen eines Befehlssatzes,
gekennzeichnet durch eine Ereignisbearbeitungseinheit (**68**), die in einer ersten Betriebsart die Bearbeitung von Ereignissen zuläßt, durch die sich der Schaltzustand des Prozessors (**50**) verändert, ohne daß die Arbeit der Steuereinheit unterbrochen wird, und die in einer zweiten Betriebsart die Bearbeitung der Ereignisse sperrt, wobei die Steuereinheit mehrere Funktionseinheiten enthält, die parallel und/oder zeitlich überlappend an verschiedenen Befehlen des Befehlssatzes oder an verschiedenen Teilen eines Befehls arbeiten und daß in der zweiten Betriebsart geprüft wird, ob ein Ereignis zu bearbeiten ist,
und daß abhängig vom Prüfergebnis beim Vorliegen eines Ereignisses in die erste Betriebsart geschaltet wird.

5. Prozessor (**50**) nach Anspruch 4, dadurch gekennzeichnet, daß die Betriebsart durch das Ändern eines Statusbits (**64**) in einem Statuswort (**60**) zur Steuerung der Arbeitsweise des Prozessors (**50**) auswählbar ist.

6. Prozessor (**50**) nach einem der Ansprüche 4 bis 5, dadurch gekennzeichnet, daß die Ereignisse durch prozessorexterne Vorgänge (**416**, **418**) hervorgerufen werden.

7. Prozessor (**50**) nach einem der Ansprüche 4 bis 6, gekennzeichnet durch eine Speichereinheit (**68**) zum Speichern von Ereignismeldungen beim Auftreten von Ereignissen in der zweiten Betriebsart.

8. Prozessor (**50**) nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß der Prüfschritt durch einen speziellen Befehl des Befehlssatzes auslösbar ist.

9. Prozessor (**50**) nach Anspruch 8, dadurch gekennzeichnet, daß der Prüfschritt vor und/oder nach der Ausführung des speziellen Befehls ausgelöst wird.

10. Prozessor (**50**) nach einem der Ansprüche 1 bis 3 und einem der Ansprüche 4 bis 9 sowie nach Anspruch 10 oder 11, dadurch gekennzeichnet, daß der spezielle Befehl sowohl den Prüfschritt in der Unterbrechungsbearbeitungseinheit als auch den Prüfschritt in der Ereignisbearbeitungseinheit auslöst.

11. Prozessor (**50**) nach einem der Ansprüche 1 bis 7, dadurch gekennzeichnet, daß der Prüfschritt durch einen Indikator (**210**; **310**) in mindestens einem Befehlswort (**200**; **300**) ausgelöst wird.

12. Prozessor (**50**) nach Anspruch 11, dadurch gekennzeichnet, daß bei gesetztem Indikator (**210**; **310**) der Prüfschritt vor und/oder nach der Ausführung des Befehls (**202** bis **206**, **302** bis **306**) ausgelöst wird, der durch das den Indikator (**210**; **310**) enthaltende Befehlswort (**200**; **300**) festgelegt ist.

13. Prozessor (**50**) nach Anspruch 11 oder 12, dadurch gekennzeichnet, daß das Befehlswort (**200**; **300**) mehrere Befehle (**202** bis **206**, **302** bis **306**) enthält.

14. Prozessor (**50**) nach einem der Ansprüche 1 bis 3 und einem der Ansprüche 4 bis 9 sowie einem der Ansprüche 13 bis 15, dadurch gekennzeichnet, daß der Indikator sowohl den Prüfschritt in der Unterbrechungsbearbeitungseinheit als auch den Prüfschritt in der Ereignisbearbeitungseinheit auslöst.

15. Prozessor (**50**) nach einem der Ansprüche 1 bis 3 und einem der Ansprüche 4 bis 9 sowie einem der Ansprüche 13 bis 15, dadurch gekennzeichnet, daß das Befehlswort für die Unterbrechungsbearbeitungseinheit und die Ereignisbearbeitungseinheit verschiedene Indikatoren enthält.

16. Prozessor (**50**) nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß der Prozessor (**50**) bei der Emulation eines Prozessors anderer Bauart verwendet wird.

17. Prozessor (**400**, **402**) nach einem der vorhergehenden Ansprüche, dadurch gekennzeichnet, daß der Prozessor (**400**, **402**) in einem Mehrprozessorsystem (**400**, **402**) eingesetzt wird.

Hierzu 5 Seite(n) Zeichnungen

Nummer: DE 100 08 008 C1
Int. Cl.[7]: G 06 F 9/38
Veröffentlichungstag: 23. August 2001

FIG 1

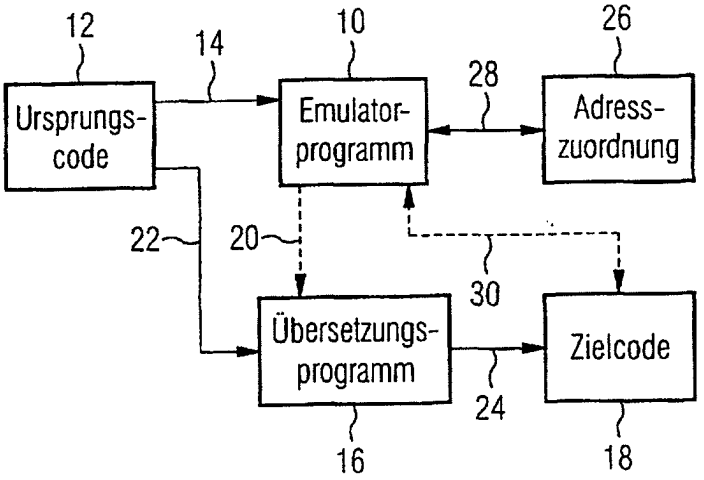
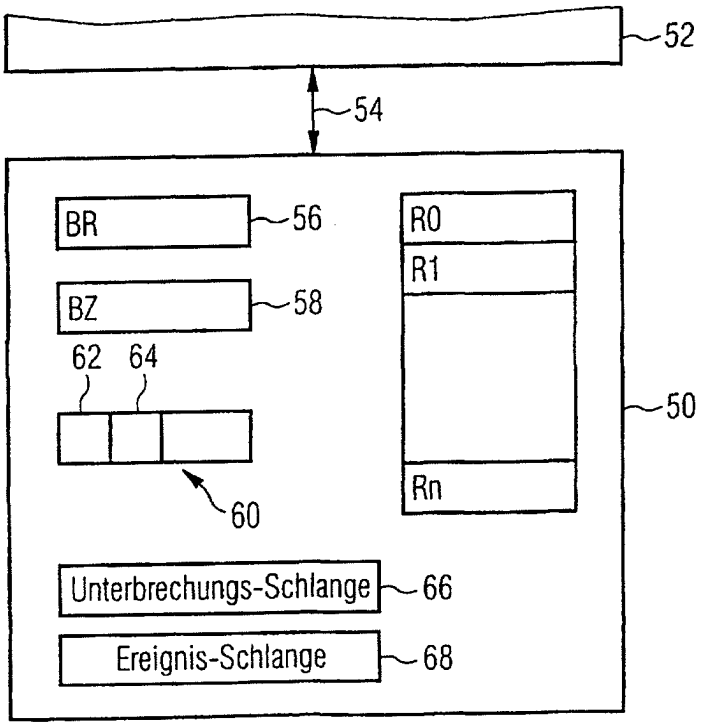


FIG 2



Nummer: DE 100 08 008 C1
Int. Cl.[7]: G 06 F 9/38
Veröffentlichungstag: 23. August 2001

FIG 3

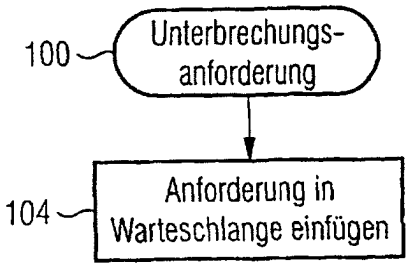
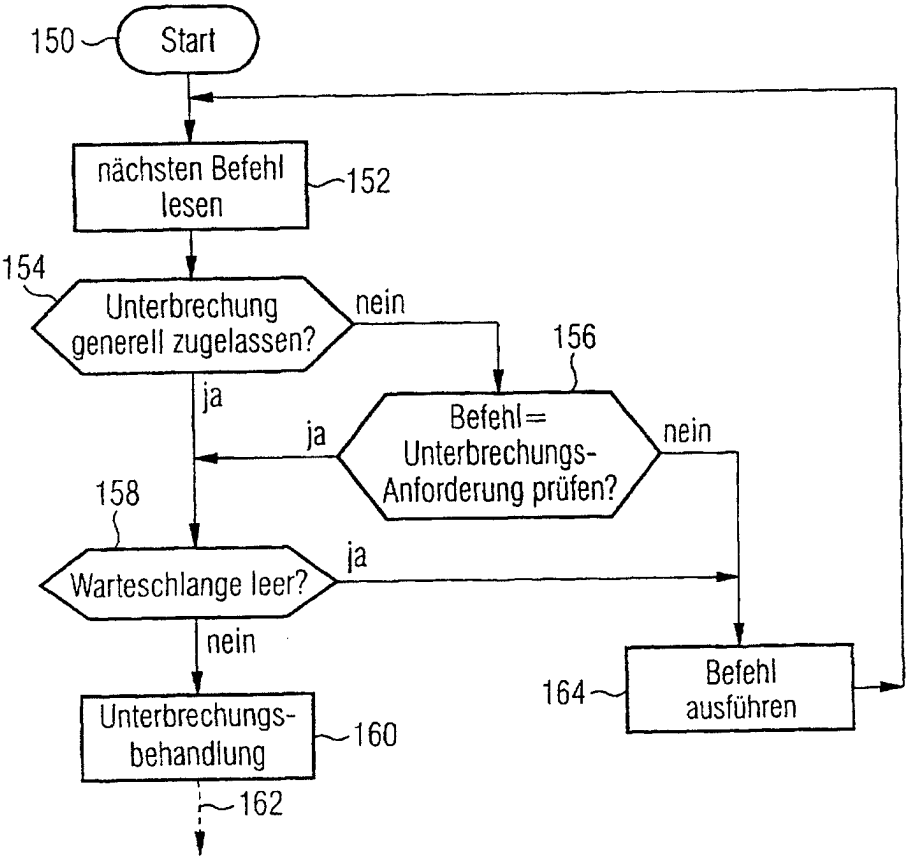


FIG 4



Nummer: DE 100 08 008 C1
Int. Cl.[7]: G 06 F 9/38
Veröffentlichungstag: 23. August 2001

FIG 5

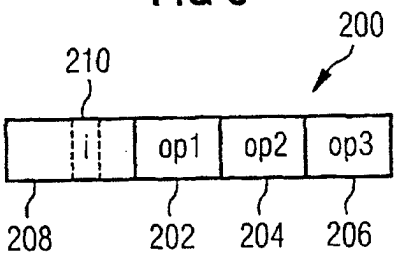
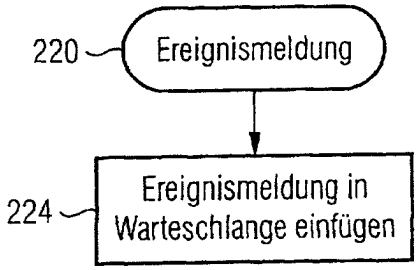


FIG 6



Nummer: DE 100 08 008 C1
Int. Cl.[7]: G 06 F 9/38
Veröffentlichungstag: 23. August 2001

FIG 7

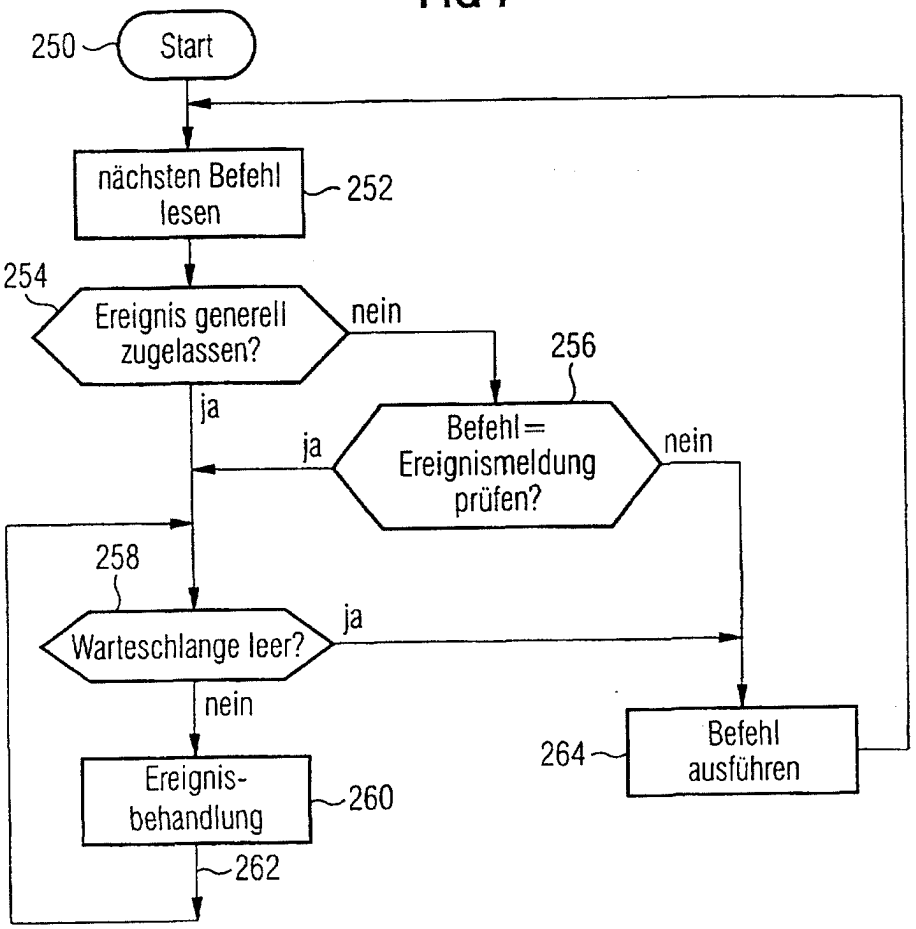
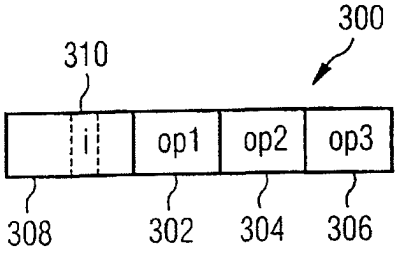


FIG 8



Nummer: DE 100 08 008 C1
Int. Cl.[7]: G 06 F 9/38
Veröffentlichungstag: 23. August 2001

FIG 9